Predicting user satisfaction to optimize AP selection in WLANs using Random Forests

Marc Carrascosa Zamacois

Master thesis

Master in Intelligent Interactive Systems

Escola superior politècnica UPF

2019

Thesis director

Boris Bellalta Jiménez

Acknowledgements

I want to thank my supervisor Boris Bellalta for all the support he has given me.

I would also like to thank everyone in the Wireless Networking Research Group for all their help, both with the master and the office rodents.

Finally, the biggest thank you to my parents for standing by me all these years.

This work has been partially supported by a Gift from the Cisco University Research Program (CG#890107, Towards Deterministic Channel Access in High-Density WLANs) Fund, a corporate advised fund of Silicon Valley Community Foundation.

Abstract

Nowadays, it is common to find WiFi networks that have a central controller connected to all Access Points in the network to both organize them and collect relevant information from them. This creates huge amounts of data which can open new avenues for Machine Learning to be used in wireless networks, as the amount of data can be impossible to parse by a human. In this work, we propose a Supervised Learning model based on Random Forests that can parse all this data and allow us to predict the satisfaction of all users in the network. To study its performance, we create a simulated environment from which we can extract a data set to train the model. Afterwards, we use this model to analyze the importance of the metrics available and test it in the simulator to confirm its effectiveness. We then use the same model to create a process in the simulated central controller that can reassociate users to Access Points that will offer a better service, reaching a higher network performance and increasing average user satisfaction.

Resum

Avui dia és típic que una xarxa WiFi tingui una controladora central connectada a tots els punts d'accés de la xarxa, tant per configurar-los com per recollir informació rellevant de la seva activitat. Aquests processos creen grans quantitats de dades que ofereixen noves possibilitats per la utilització de Machine Learning en xarxes sense fils, ja que la quantitat d'informació generada pot ser impossible de processar per un humà. En aquest document proposem un model de Supervised Learning basat en Random Forests que ens permetrà predir la satisfacció de tots els usuaris en una xarxa. Hem creat una plataforma de simulació de la qual extraiem un data set amb el qual realitzar l'estudi. Un cop tenim el model, l'utilitzem per a analitzar les mètriques més importants d'una xarxa i el testem en la simulació per a confirmar la seva efectivitat. Finalment, utilitzem aquest model per a crear un procés a la controladora central que reassocii usuaris a punts d'accés que puguin oferir un millor servei, obtenint un major rendiment de la xarxa i incrementant la satisfacció mitjana per usuari.

Resumen

Hoy día es habitual que una red WiFi tenga una controladora central conectada a todos los puntos de acceso de la red, tanto como para configurarlos como para recoger información relevante de su actividad. Estos procesos crean grandes cantidades de información que ofrecen nuevas posibilidades de utilizar Machine Learning en redes inalámbricas, ya que la cantidad de información generada puede ser imposible de procesar por un humano. En este documento proponemos un modelo de Supervised Learning basado en Random Forests que nos permitirá predecir el nivel de satisfacción de todos los usuarios de la red. Hemos creado una plataforma de simulación de la cual extraemos un data set con el cual realizaremos el estudio. Una vez tenemos el modelo, lo usamos para analizar las métricas más importantes de una red y lo testeamos en la simulación para confirmar su efectividad. Finalmente, utilizamos este modelo para crear un proceso en la controladora central que reasocie a usuarios a puntos de acceso que puedan ofrecer un mejor servicio, obteniendo un mayor rendimiento en la red e incrementando la satisfacción media por usuario.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

WiFi grows more popular every day, with users wanting Internet access wherever they go. WiFi connections represented 43% of all traffic in 2017, but this number is expected to increase to 51%, half of all IP traffic, by 2022 [1]. The amount of Access Points (APs) will also increase to keep up with this demand, with the total amount of public WiFi hotspots going from 124 million in 2017 to 549 million in 2022. The number of connected devices is also constantly increasing, with 2.4 networked devices per person in 2017, and a projected 3.6 in 2022 [2]. Furthermore, users require higher data rates and higher amounts of traffic than ever before.

All these demands have lead to the densification of networks, meaning that the number of APs per squared meter has increased. This allows each AP to serve a lower amount of users, as well as offer them a higher data rate. Current efforts in the IEEE 802.11 standard are also building towards measures to cope with the higher demands. The 802.11ac amendment already brought the 256-QAM modulation for a higher data rate, as well as beamforming and downlink MU-MIMO to increase the signal power received by end users and allow APs to serve multiple users at once. The 802.11ax amendment will bring even higher speeds through 1024-QAM, as well as uplink MU-MIMO to allow multiple users to transmit data at the same time [3].

In this kind of dense deployments, it is common to find multiple overlapping APs, and any change in the configuration is expected to entail a higher impact than in sparser networks in which APs can operate in isolation. This sets up new challenges when configuring the network, as overlapping APs will interfere with each other unless their channel selection is not properly planned. Interference can also be avoided by adjusting the power transmission so that the coverage areas of APs are not overlapping. This higher AP density also creates new opportunities, as user stations (STAs) are in range of multiple APs, allowing the offloading of STAs from one AP to another to balance their loads. These decisions can be made by the AP, as they can disassociate users if their load gets too high, but they are usually relegated to a central controller that is connected to all APs in the network, thus obtaining a

global view of the whole network, allowing it to make more informed decisions.

A simple example can be seen in Figure 1.1, where we see that STA 2 and STA 4 are in range of two APs but decide to associate to AP 2 due to their stronger signal, thus leading to AP 3 and AP 1 being underutilized. The central controller can see this disparity in loads and forward a request to STA 2 in order to re-associate to AP 1. The same can be done for STA 4, which can be re-associated to AP 3, thus spreading the STAs evenly and reducing the load on AP 2.

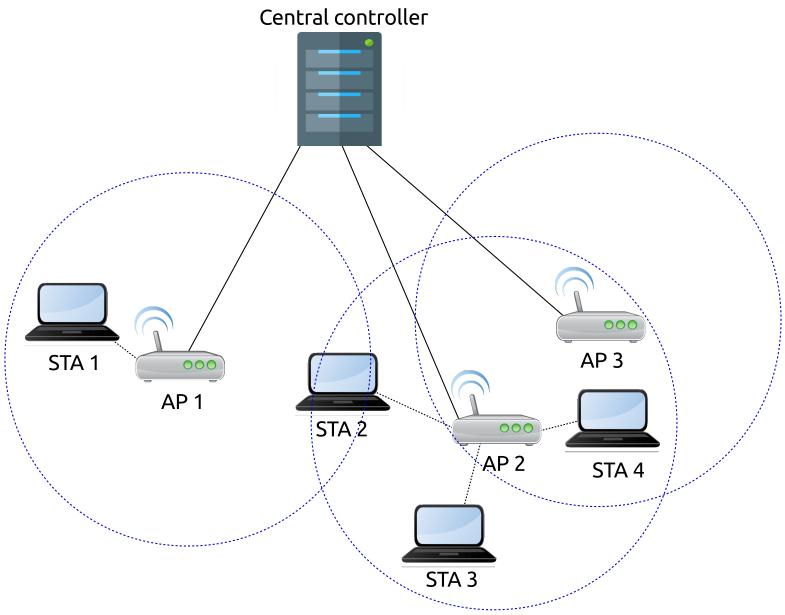


Figure 1.1: Example of considered scenario

With all this information being gathered by the controller, the decision making becomes more complicated, and we believe that machine learning algorithms can help process all this information and help in making decisions that allow us to optimize the network. We intend to use Supervised Learning methods specifically to take advantage of all the logs recovered by the controller and create a user association mechanism that improves overall network satisfaction.

1.2 Contributions

We aim to improve IEEE 802.11 WLANs by re-associating STAs to APs that will give them a better service. More specifically we will:

1. Create a simulator tool using C++ that captures the usual behavior of IEEE 802.11 WLANs, especially for multi-AP scenarios. It needs to be capable of performing large simulations efficiently, as it will be used to create a large data set to analyze through Machine Learning.

2. Study Supervised Learning techniques that can help us to create a model to predict user satisfaction in the network.
3. Use the aforementioned model to find the best performance metrics that should be considered when analyzing user satisfaction in a network.
4. Implement the Supervised Learning model and evaluate its performance in a WLAN through simulation.
5. Create an algorithm that can be used by a network controller to find the optimal AP-STA association.

1.3 Document structure

This document is organized as follows: in Chapter 2 the state of the art is presented for both AP selection efforts, as well as machine learning solutions to network configuration. Chapter 3 summarizes the system model of our simulator, as well as the tools used to develop it. Chapter 4 presents our Supervised Learning model, the metrics used and its performance in a simulation. In Chapter 5 the AP selection algorithm is implemented and tested in our simulator. Finally, some conclusions are discussed in Chapter 6.

Chapter 2

Related Work

2.1 Wireless Networks and Machine Learning

Machine Learning is now more popular than ever, and their use is being studied in many fields. Wireless network optimization is one such application, and there is plenty of literature for WiFi as well as cellular networks and Wireless Sensor Networks.

The authors in [4] study the benefits of Machine Learning in 5G wireless networks and give a short description of most popular types of Learning: Supervised, Unsupervised and Reinforcement Learning, including the Multi-Armed Bandits problem. A longer and more detailed study on Cognitive Radios can be found in [5], including a section on distributed and centralized learning, commenting on the trade-offs involved with using one or the other.

The authors in [6] use several Multi-Armed Bandits algorithms to optimize the channel selection and transmission power used by multiple WLANs in the same area so as to maximize the throughput achieved by each of them. A comparison of these algorithms is made, showing Thompson Sampling to be very suited to this decentralized approach, obtaining fairness and avoiding high throughput variability.

In [7], the authors use a Support Vector Machine to predict a user's dwell time on a coffee shop using the RSSI, the transmission rate and accelerometer information from their phone. Then this information is used to give priority to the downloads of users that are leaving the WiFi range so that the traffic offloaded to 3G can be minimized.

In [8] a model is proposed using both SVM and regression analysis to predict the effect of a channel change in an AP using only AP traffic and RSSI. The SVM is trained to find if the channel change would lead to saturation, and the regression is used to predict the throughput and delay obtained for the saturated cases.

The work in [9] uses a decision tree to steer STAs to WiFi or 3G networks depending on their RSSI, speed, location and type of connection to provide the highest Quality of Service and lowest energy consumption.

2.2 The problem of AP selection

If we look specifically at the AP selection problem we can find that it has been studied extensively. This is due to the way that standard association works in IEEE 802.11 networks. This method has the STAs scan all available channels for available APs and associate to the one with the highest Received Signal Strength Indicator (RSSI). This method leads to uneven loads on APs, as the number of users on each AP depends on the AP location, as well as the user behaviour [10] [11].

Earlier works that did not use Machine Learning used the delay between AP beacon transmissions to estimate the available throughput of an AP [12] or proposed modifications of probe and beacon fields to inform new users of the AP load so that they could make an estimation based on their received signal and available data rate [13].

Currently, a lot of the study has moved towards using Machine Learning techniques. The authors in [14] use decision trees to predict if a user will have a high or low latency in a particular AP based on their SNR, the channel utilization and the number of devices connected. A phone app was built to detect nearby APs and inform the central controller, which then uses this decision tree to tell the user to re-associate to an AP that will offer a faster connection, obtaining a favorable result in 93% of re-associations.

The model proposed in [15] uses random forests to predict the time required to associate to an AP by classifying all available APs sensed by the STA as fast or slow and then taking the one with the higher RSSI in the fast class. This reduces the connection failures from 33% to 3.6% and reduces the connection time by a factor of ten.

The work in [16] uses a decentralized approach in which STAs are equipped with a neural network that uses the Signal to Noise Ratio (SNR), the number of retransmitted frames, the amount of time that the channel is busy and the number of detected STAs to estimate the throughput obtainable on each available AP and picking the highest one for association.

Our work in [17] presents a decentralized system for AP selection using Multi-Armed Bandits and ε-greedy. In it, STAs explore all APs available to them at regular intervals until satisfaction is found and then the STA sticks to the current AP unless the network configuration changes and the satisfaction decreases.

Chapter 3

Simulator

3.1 Design principles

In this Chapter we will describe the tools used to create the simulator, as well as its characteristics, assumptions made and parameters required for a simulation.

Our simulator was programmed in C++ using the CompC++ and COST libraries[1]. The first one allows us to use classes as components that can be connected to one another through inports and outports, and the second one is a sequential simulation engine which is designed to use classes that are aware of the simulation time, letting us trigger events based on it.

3.2 System model

We create a 3D environment with M APs and N STAs placed randomly following a uniform distribution. After that, each STA associates to the AP with the strongest signal available, thus following the standard procedure. STAs follow an ON/OFF activity model. When they are active, we consider they require a throughput of w Mbps. ON and OFF periods have a random duration following an exponential distribution. All APs operate in the 2.4 GHz band using only orthogonal channels of 20 MHz, meaning that only channels 1, 6 and 11 are available. The channel allocated to each AP is chosen uniformly at random. Both APs and STAs transmit at 20 dBm and follow the 802.11n standard in terms of data rate.

3.2.a Path-loss

We use the path-loss model from the 802.11ax task group simulation scenarios [18], specifically, the residential scenario, as it is the one that considers multiple floor

[1] `http://www.ita.cs.rpi.edu/`

environments. It is defined as:

$$\begin{aligned} \mathrm{PL(d)} = \mathrm{PL}_0 + 20\log_{10}\left(\frac{f_c}{2.4}\right) + 20\log_{10}(\min(d_{i,j},5)) + \\ +(d_{i,j}>5)\cdot 35\log_{10}\left(\frac{d_{i,j}}{5}\right) + 18.3\cdot F^{\left(\frac{F+2}{F+1}-0.46\right)} + 5\cdot\left(\frac{d_{i,j}}{d_w}\right) \end{aligned} \tag{3.1}$$

where PL_0 is the path-loss at one meter, $d_{i,j}$ is the distance between STA i and AP j, f_c is the central frequency of the band used, d_w is the average distance between walls and F is the average number of floors traversed, calculated as:

$$F = \left|\frac{Z_{\mathrm{STA}} - Z_{\mathrm{AP}}}{Z_{\mathrm{ceiling}}}\right| \tag{3.2}$$

where Z_{STA} and Z_{AP} are the z coordinates for the STA and AP, and Z_{ceiling} is the height of the ceiling. The values of these parameters are summarized in Table 3.1.

Parameter	**Value**
Area(m)	50×50
PL_0(dB)	40.05
Frequency Channels available	{1, 6, 11}
Channel bandwidth (MHz)	20
Spatial streams	2
f_c (GHz)	2.4
d_w(m)	5
Z_{ceiling} (m)	3
AP transmission power (dBm)	20
STA transmission power (dBm)	20

Table 3.1: Simulation parameters

3.2.b Airtime model

The airtime required by each STA is the fraction of time required when it is active. It is calculated after every change in the network, using the required throughput w of the STA and its transmission rate r, which we obtain through the path-loss calculation previously described. The parameters used for this section and their values can be found in Table 3.2.

We start with the transmission time for a data frame, which is given by:

$$T(L,r) = T_{\text{data}}(L,r) + \text{SIFS} + T_{\text{ack}}(r) + \text{DIFS} + T_e \tag{3.3}$$

where

$$T_{\text{data}}(r) = T_{\text{PHY}} + \left\lceil \frac{L_{\text{SF}} + L_{\text{MH}} + L_i + L_{\text{TB}}}{r} \right\rceil \sigma \tag{3.4}$$

and

$$T_{\text{ack}}(r) = T_{\text{PHY-legacy}} + \left\lceil \frac{L_{\text{SF}} + L_{\text{ACK}} + L_{\text{TB}}}{r} \right\rceil \sigma \tag{3.5}$$

Then, the airtime required by STA i, including the average back-off period, is given by:

$$\alpha_i(w,L,r) = \frac{w}{L} \cdot (E[\psi]T_e + T(L,r)) \tag{3.6}$$

Next we check the network capacity by adding the airtime of every STA in the coverage area of the AP that is using the same channel:

$$\alpha_j(w,L,r) = \min(1, \sum_{\forall i \in U_j} \alpha_i(w,L,r)) \tag{3.7}$$

where U_j is the set of STAs in the coverage range of AP_j, even if not associated to AP_j but using the same channel.

Finally, the actual airtime received by the STAs is set as:

$$\beta_i = \frac{\alpha_i(w,L,r)}{\max(1, \alpha_j(w,L,r))} \tag{3.8}$$

Meaning that if $\alpha_j(w,L,r) \leq 1$ the STA receives all the airtime needed, and if it is higher than 1 it receives a proportional fraction.

Description	Parameter	Value
Preamble duration	T_{PHY}	$40\mu s$
Legacy preamble duration	$T_{\text{PHY-Legacy}}$	$20\mu s$
OFDM symbol duration	σ	$4\mu s$
Short InterFrame Space	SIFS	$10\mu s$
DCF InterFrame Space	DIFS	$50\mu s$
Average back-off duration	$E[\psi]$	7.5 slots
Empty backoff slot duration	T_e	$20\mu s$
Service Field length	L_{SF}	16 bits
MAC header length	L_{MH}	240 bits
Tail length	L_{TB}	6 bits
ACK length	L_{ACK}	112 bits
Frame size	L	12000 bits

Table 3.2: Parameters used

3.3 Simulation examples

We show the type of scenarios that can be created in Figure 3.1, where we generate a building with 2 floors and 4 apartments of $20 \times 20 \times 3$ metres, with 2 APs and 10 STAs per apartment.

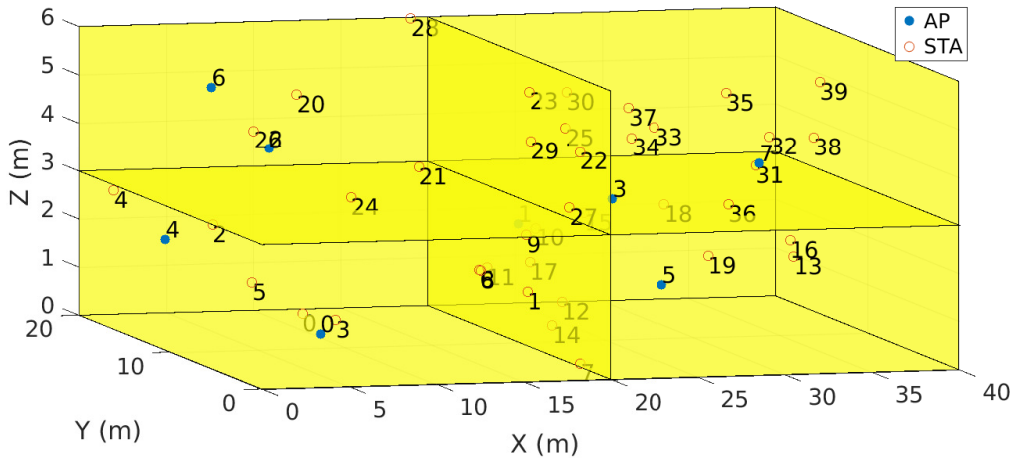


Figure 3.1: Residential scenario

Figure 3.2 shows the channel occupation for an entire hour in all APs. APs with the same color share the channel. It can be seen that AP 5 and AP 7 are in range of each other, as they have the same load, while AP 6 is far enough not to sense them, thus having a different load despite being in the same channel. We can also see that users in AP 2 or 4 will suffer starvation, as their APs are overloaded, while users in AP 3 or 6 will receive all their requested throughput properly.

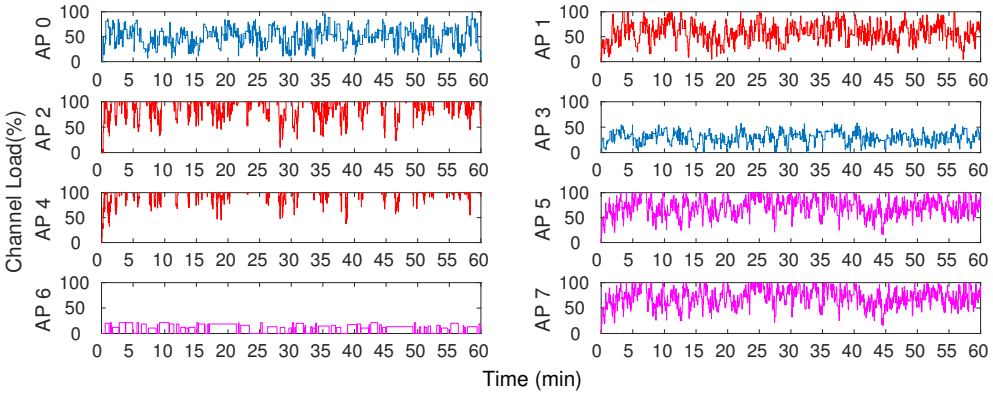


Figure 3.2: Channel activity for two APs

Chapter 4

Predicting STA satisfaction with Supervised Learning

4.1 Introduction

In this Chapter we will model the satisfaction of a STA through Supervised Learning. We will define this satisfaction as the STA receiving all of its requested throughput. We will use several simulations to obtain a data set and analyze it with multiple Supervised Learning algorithms, including an analysis of which parameters obtained by the central controller are more important for the decision making. Finally, we will implement the model in our simulation to test its efficacy.

4.2 Building the data set

The first thing that needs to be defined is what we want to predict. We want to know if a STA will be satisfied (satisfaction is defined in equation 3.8 as the fraction of airtime received by the STA vs. the required one) for the duration of an hour. In our model, we compute this received airtime for every active period of a STA and we can then obtain the average satisfaction over a period of time.

Satisfaction can be in the range of $[0, 1]$ and we will consider a STA to be satisfied if its average satisfaction is 1. With this we define two classes of STA: those that are satisfied and those that are not, which we define as those that receive less than 100% of their requested airtime at any point in the simulation time.

Next, we define the features that can be extracted from the STAs and their APs that can help us to create a model that predicts the classes previously mentioned. The features available to us are:

1. **RSSI :** Received Signal Strength Indicator for the STA - AP link (dBm).
2. $\mathbf{N_{STA}}$**:** The number of STAs connected to the associated AP.
3. $\mathbf{N_{AP}}$**:** The amount of APs in the same channel as the associated AP.
4. $\mathbf{N_{AP\text{-}STA}}$**:** The number of STAs connected to APs that are both in the same channel as the AP of the STA and in range of it. Calculated at the controller by adding the information received from all APs in the network.
5. $\mathbf{L_{avg}}$**:** The average channel load perceived by the AP.
6. $\mathbf{S_{avg}}$**:** The average throughput w requested by the STA for each transmission multiplied by the duration of the time the transmission lasts.
7. $\boldsymbol{\alpha_{avg}}$**:** The average airtime requested.

Most of the features have different ranges, which are specified in Table 4.1. Those of them that have a range of up to ∞ will be limited by our simulation scenarios. For instance, $\mathrm{N_{STA}}$ will not be higher than the amount N of STAs that we simulate, and $\mathrm{N_{AP}}$ will have a maximum defined by M. RSSI starts at -82 as it is the default Clear Channel Assessment (CCA) for WiFi, and its maximum is 20 dBm, which corresponds to the transmission power received if there were no losses for that link (and because we consider 20 dBm as the transmission power for APs).

Feature	**Source**	**Range**
RSSI	STA	$\mathbb{R} \in [-82, 20]$
$\mathrm{N_{STA}}$	AP	$\mathbb{N} \in [1, \infty)$
$\mathrm{S_{avg}}$	STA	$\mathbb{N} \in [1, \infty)$
$\mathrm{N_{AP\text{-}STA}}$	AP/Controller	$\mathbb{N} \in [1, \infty)$
$\mathrm{N_{AP}}$	AP	$\mathbb{W} \in (0, \infty)$
$\mathrm{L_{avg}}$	AP	$\mathbb{R} \in (0, \infty)$
α_{avg}	STA	$\mathbb{R} \in (0, \infty)$

Table 4.1: Feature range

The importance of these ranges lies in the fact that they are different, which will create issues when using Supervised Learning algorithms, as they usually work by tuning the features of a dot product of the form:

$$\mathrm{coeff}_0 \cdot \mathrm{feature}_0 + \mathrm{coeff}_1 \cdot \mathrm{feature}_1 + ... + \mathrm{coeff}_N \cdot \mathrm{feature}_N \tag{4.1}$$

Thus, if features work in different ranges their value is disproportionate in regards to the other features, and this can lead to problems during classification.

To fix this issue we will standardize the features during training, this means applying:

$$\text{Standardized sample}_i = \frac{\text{sample}_i - \mu_j}{\sigma_j} \tag{4.2}$$

where μ_j is the mean of the entire set of samples of feature j and σ_j is the standard deviation of the feature j. This will be applied during the training phase, and then the μ_j and σ_j of the training set are used to standardize the testing data.

Another thing to consider is that the data set needs to be balanced, requiring every class to have a similar amount of samples. This avoids bias in the model, as a training set that consists of 90% of samples of one class will lead to a model that leans towards that class for predictions. For us, this means that whenever we create a data set we will count the number of samples of each class, use all of the samples of the class with a smaller number and then draw an equal amount of samples randomly from the bigger set of samples of the other class.

For the generation of samples we will run 500 simulations that will last one hour each. Then we will extract all the previously mentioned features for each STA that had an association. We will also extract the satisfaction and according to the previously mentioned classification we create two classes, class 0 for unsatisfied STAs and class 1 for satisfied STAs. We show an example of the data set in Figure 4.2, both for the regular data and its standardized form.[1]

Regular data set							
RSSI	$\mathbf{N_{STA}}$	$\mathbf{N_{AP}}$	$\mathbf{N_{AP\text{-}STA}}$	$\mathbf{L_{avg}}$	$\mathbf{S_{avg}}$	$\alpha_{\mathbf{avg}}$	**Class**
−50.9565	9	1	12	0.8383	2.8686	0.0554	0
−81.6883	6	1	14	0.9296	3.1814	0.4036	0
−63.0984	3	1	12	0.8355	2.2424	0.0471	0
−79.6619	6	0	6	0.3588	2.5342	0.3373	1
−74.6483	7	0	7	0.3620	2.5436	0.1285	1
−58.5088	6	0	6	0.3588	2.6818	0.0489	1
Standardized data set							
0.6557	0.5783	1.2414	1.4921	0.5419	−0.4184	0.7943	0
−1.6731	−0.4673	1.2414	1.8496	1.6162	4.2599	1.3104	0
−0.2644	−1.5129	1.2414	1.4811	−1.6089	−0.5297	0.7943	0
−1.5195	−0.4673	−0.6034	−0.3861	−0.6066	3.3699	−0.7539	1
−1.1396	−0.1188	−0.6034	−0.3735	−0.5745	0.5635	−0.4959	1
0.0834	−0.4673	−0.6034	−0.3861	−0.0996	−0.5057	−0.7539	1

Table 4.2: Example of data set

[1]The data sets used can be found at: `https://github.com/MCarrascosaZ/TFMDataSets`

4.3 Model selection

Here we will use different methods to test our data set and find an accurate prediction model. We will be using Python and the scikit learn package [2] for machine learning.

Our first simulation is done for an area of $40x40x3$ metres with 3 APs and 20 STAs. All STAs request 1 to 8 Mbps uniformly, the time a flow is active is an exponentially distributed random variable with mean 30 seconds, and the time between activity is also an exponentially distributed random variable but with mean 20 seconds. STAs only have one flow active at a time. The reason for these particular parameters can be seen in Figure 4.1, which shows that most STAs are satisfied, and that the unsatisfied STAs are really close to being satisfied, with some of them having a satisfaction ratio as high as 99.998%. These should be the more complicated cases to predict, and with 500 simulations using different seeds we get 4447 STAs with satisfaction equal to 1 and 4181 unsatisfied STAs (satisfaction less than 1).

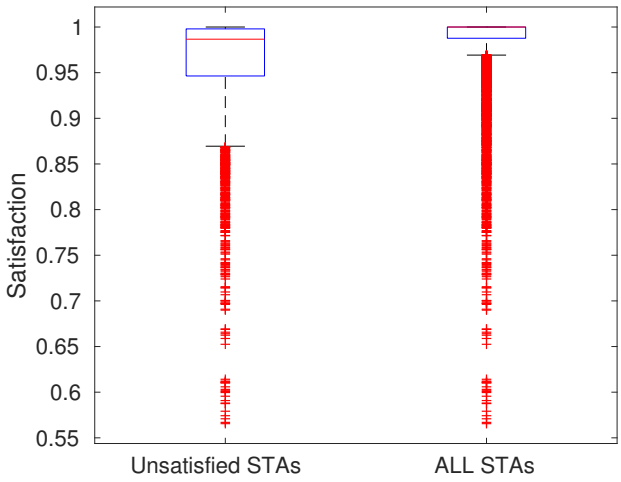
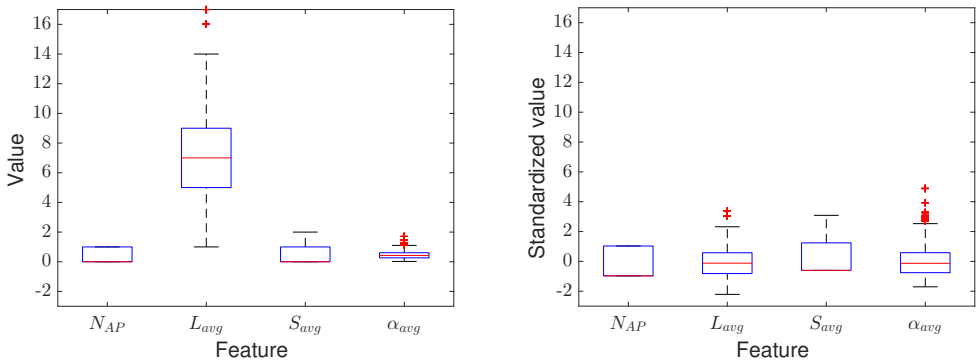


Figure 4.1: Satisfaction values

We also show some of the feature ranges in Figure 4.2a, as well as their standardized range in 4.2b to show the effect of our preprocessing.

[2] `https://scikit-learn.org/stable/`



(a) No preprocessing (b) Standardized

Figure 4.2: Feature range for similar features

In the following sections we will test three classification models to both validate the possibility of proper classification and to find the optimal model for our purposes. We will use Support Vector Machines (SVM), Logistic Regression and Random Forests.

4.3.a Brief introduction to the classification methods

Support Vector Machines

Support Vector Machines are linear classifiers that treat data as vectors and try to find a hyperplane that separates the two classes with the widest possible margin. The equation of a hyperplane is:

$$b + \sum_{i=1}^{d} w_i x_i = \vec{w} \cdot \vec{x} + b = 0 \tag{4.3}$$

with b being the bias, $\vec{w}$ the vector of weights and $\vec{x}$ the vector of input features. Samples will be classified as one class or another if $\vec{w} \cdot \vec{x} + b < 0$ or $\vec{w} \cdot \vec{x} + b > 0$. The SVM then optimizes $\vec{w}$ so as to find the maximum margin between the two classes. Figure 4.3 shows this idea through a set of data points belonging to different classes that can be separated by several hyperplanes, but only one of them allows the maximum margin to avoid misclassifying samples in the future.

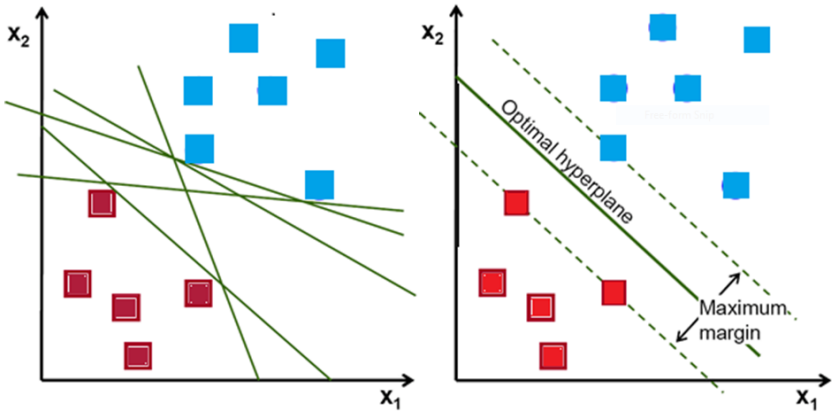


Figure 4.3: SVM margin [3]

Logistic Regression

Logistic Regression, much like SVM, is a linear classifier that separates data in two classes, in this case with a probabilistic method. It first calculates a weighted score like a linear regression:

$$S = \vec{w} \cdot \vec{x} + b \tag{4.4}$$

This is then converted to a probability by using a logistic function:

$$p(x) = \frac{1}{1 + e^{-S}} \tag{4.5}$$

A comparison between Linear and Logistic Regression is shown in Figure 4.4, where we see that the logistic function has a curved shape that helps to better represent the probability of the data point being on each class.

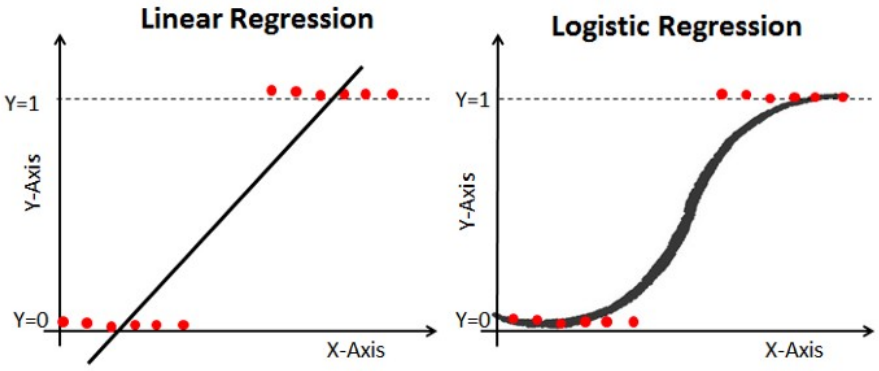


Figure 4.4: Logistic Regression [4]

[3] https://www.kraj3.com.np/2019/06/support-vector-machines-SVM-basic-concepts-and-algorithm.html
[4] https://medium.com/datadriveninvestor/logistic-regression-18afd48779ce

Random Forests and Decision Trees

Decision trees create decision graphs like the one in Figure 4.5. They take the data and try to split it into classes based on the amount of information that can be gained by the split. In this way, the higher the position of the split in the tree, the more information that is obtained by that split.

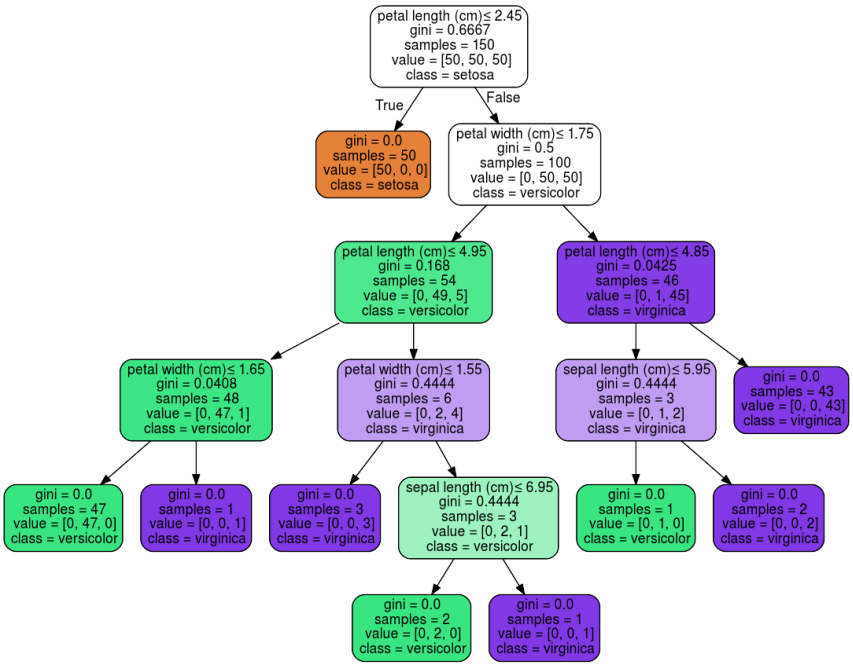


Figure 4.5: Decision Tree for flower classification [5]

A split is considered to have a high information gain if it results in two groups of samples with uneven distributions for the resulting groups of data. This is shown in Figure 4.6, where we see that if a split results in the same amount of samples for each class, then we have indeed gained no information and our ability to classify the data stays the same as before.

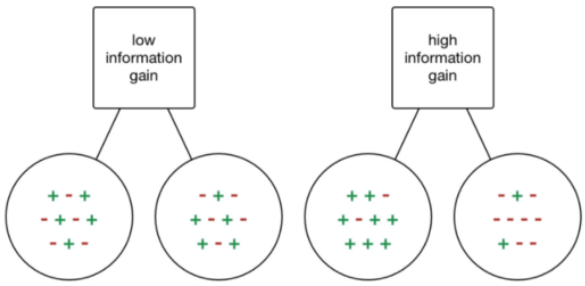


Figure 4.6: Information gain [6]

[5] https://scikit-learn.org/stable/modules/tree.html

[6] https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956

A common issue of Decision Trees is that they easily overfit. Random Forests are used to solve this issue by creating X amount of Decision Trees, randomizing the amount of features available at each split so that the trees are all different, and then using all of the trees for the classification, selecting the mode (most frequent output) of the entire forest as the final classification rule.

4.3.b Model comparison

To validate a model it is common to separate the data set in two parts, a training and a testing set. The first one is used to learn the model and the second one uses the trained model to predict the output and compare it to the actual value that should have been predicted, thus obtaining a measure of its accuracy.

Another approach is k-fold cross-validation, a method by which the training set is divided in K parts (or folds) of equal size. Then, K-1 folds are used for training and the last one for testing, going iteratively over all folds to obtain an accurate representation of the model accuracy. This allows us to first obtain several accuracy scores over the whole training set, which helps avoid bias in situations where a single testing phase is used. The process of cross-validation is shown in Figure 4.7.

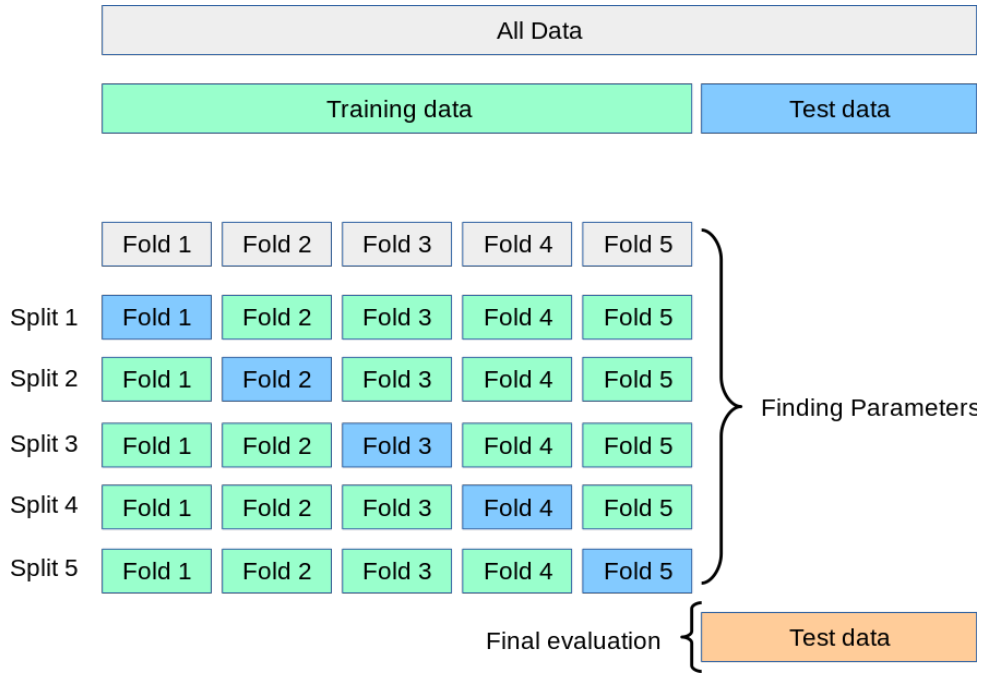


Figure 4.7: K-fold cross-validation process [7]

With cross validation we can both test the accuracy of our three chosen models, as well as see the effect of the data standardization on them. We use a 5-fold cross validation, meaning that we get 5 accuracy scores for each model. We show the obtained mean and standard deviation in the scores in Figure 4.8, where we can see

[7] `https://scikit-learn.org/stable/modules/cross_validation.html`

that SVM benefits greatly from standardized data, as its accuracy goes from 80.4% to 95.7% (a 19% increase), while the logistic regression and random forest perform very similarly in both cases, with 95.8% accuracy and 97.4% respectively.

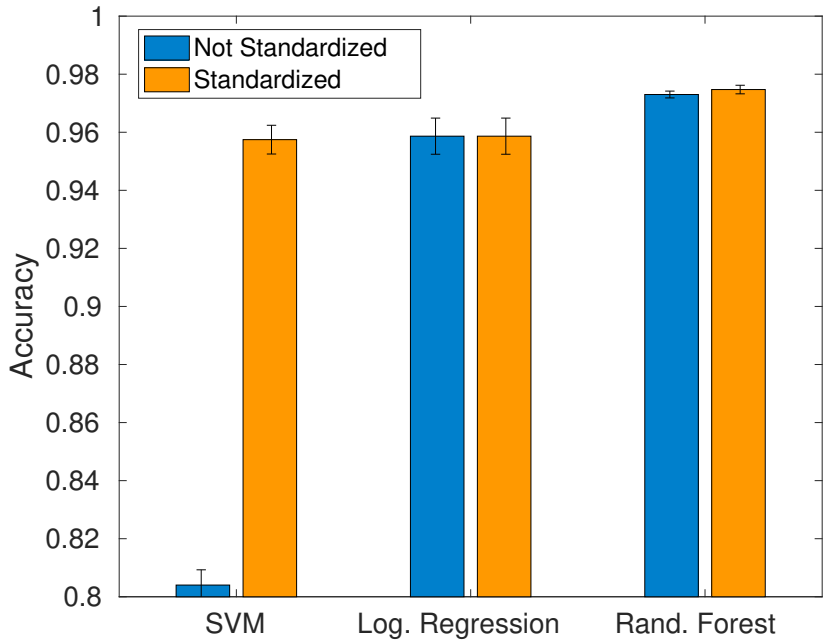


Figure 4.8: Results of the cross-validation

The need for standardization in Logistic Regression depends on the regularization applied, which tries to tune the coefficients to avoid overfitting, and, depending on the parameters used, it would lead to slightly lower results. The scaled data also helps the regression to converge faster. Decision Trees look at the entropy to perform splits in the data, which is done one feature at a time, meaning that the range of a feature is never a factor when analyzing it, and standardization is not needed.

Overall, the difference between the three models is very small, but Random Forest seems to be the better one as it outperforms both SVM and Logistic Regression in both average accuracy as well as having a slighly lower standard deviation of 0.0016 versus the 0.0062 of Logistic Regression and 0.0049 of SVM.

Finally, we fit the entire training set on all three models and test against the testing set to obtain our final testing score, and the results are very similar to those obtained through cross-validation, with Random Forest winning with a 97.6% over the 95.4% of Logistic Regression and 95.6% of SVM. Consequently, in the next sections we will use Random Forest as it seems to be the one most suited to our problem.

4.4 Feature selection

Now that we know that we can predict whether a STA will be satisfied or not, we want to understand which features are more useful for the process. One way to look at this is through the correlation between the features and the output. Figure 4.9 shows the correlation matrix of our data set. From it we can already rank the importance of the features when classifying, as the ones with a high correlation with the output used will help us classify correctly, while those that have a correlation close to 0 will not be helpful. Since we are using class 0 and 1 to represent unsatisfied and satisfied users (instead of the actual satisfaction), the sign of the correlation will not have any meaning, as it depends purely on our classification scheme (it would be the opposite if we used class 1 for unsatisfied users). For this reason we will only look at the absolute correlation values.

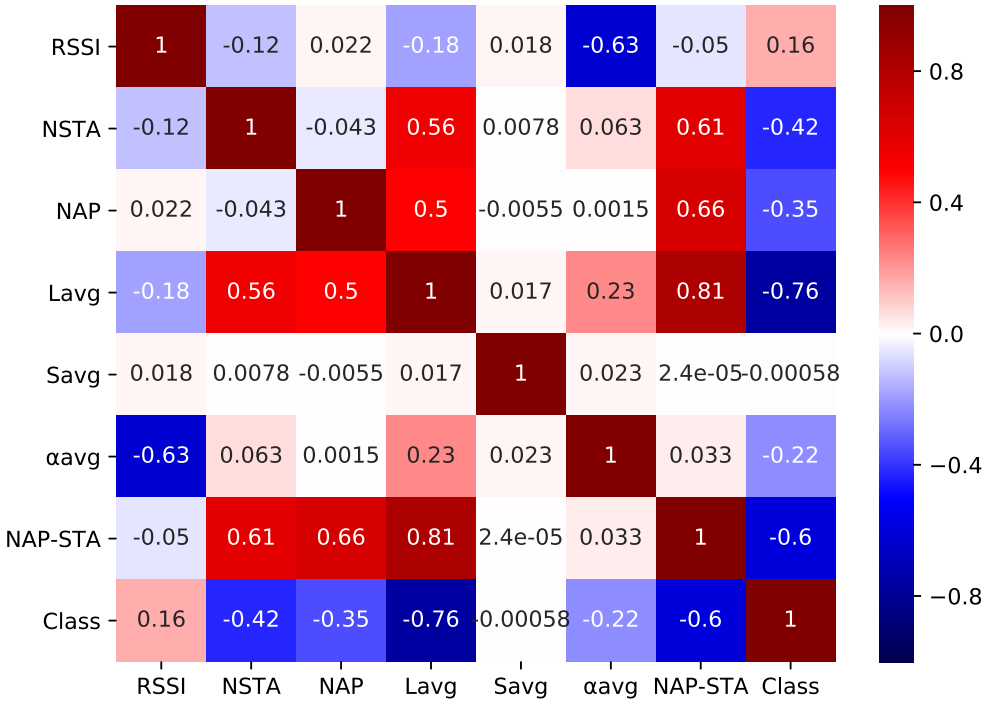


Figure 4.9: Feature correlation with class

The most important feature seems to be L_{avg}, which has a correlation of -0.76 with the class. This seems logical, as the higher the average load on the AP, the lower the user satisfaction will be. We can also find a strong correlation in $N_{AP\text{-}STA}$ with -0.6. N_{STA} and N_{AP} have a similar -0.42 and -0.35, α_{avg} is already pretty low with -0.22 and the lowest two are RSSI and S_{avg}.

We can also see that while most features have some form of correlation between themselves, S_{avg} has a correlation close to 0 with every other value in the matrix. This could be due to the variable using the requested traffic load, not the actual one received. It seems that α_{avg} should be able to replace it completely, as the airtime requested depends on both the traffic load and the RSSI, containing more information (we can see that α_{avg} and RSSI have a correlation of -0.63). But what we can also find is that the correlation between both α_{avg} and RSSI with the class is actually really similar, which could mean that the RSSI is good enough to replace those two features.

The highest value in the matrix is between L_{avg} and $N_{AP\text{-}STA}$. This could mean that their information is redundant, as they both seem to have similar correlations with N_{STA} and N_{AP}.

If we had looked only at the correlation with the output we would assume L_{avg} and $N_{AP\text{-}STA}$ to be the most important features, but seeing as they are closely related, we might want to consider N_{STA} as the second most important feature.

Before we eliminate any features however, we can also check the feature importance through the Random Forest algorithm. In it, feature importance can be extracted by looking at which branch level does the feature most often land. By checking through our 100 trees we can get a good idea of how each feature is ranked. The average importance of each feature is shown in Figure 4.10 along with its standard deviation.

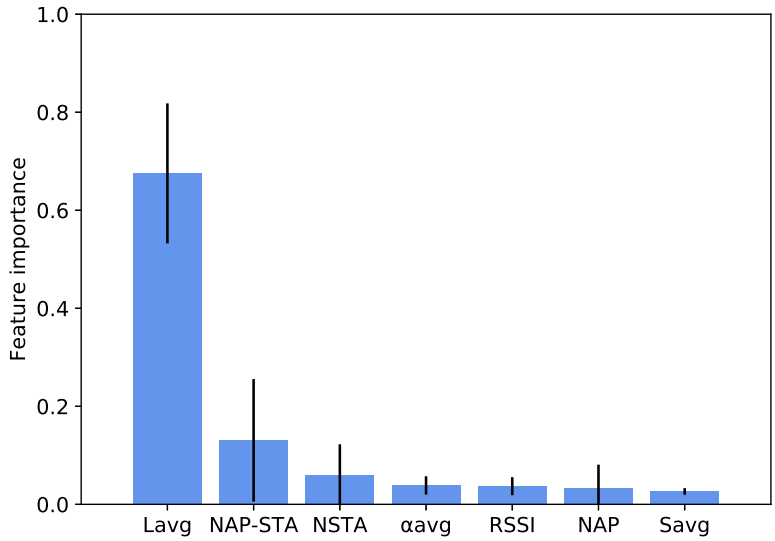
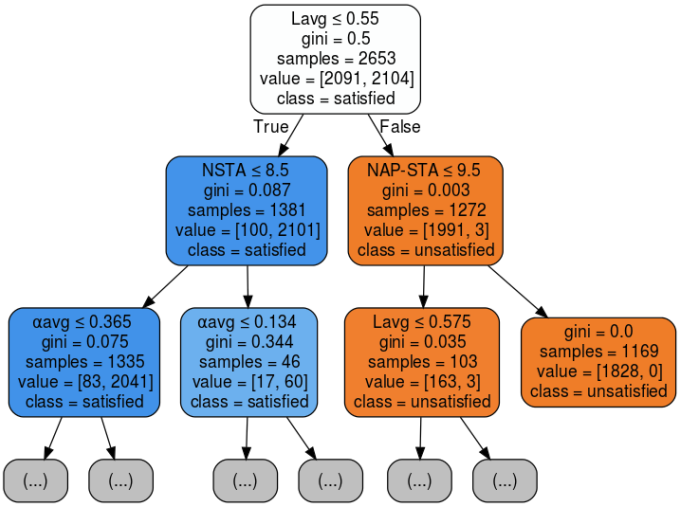


Figure 4.10: Feature importance in Random Forest

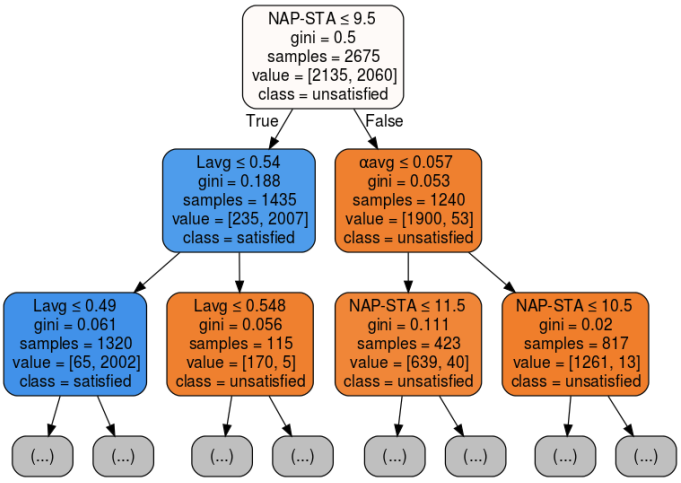
We find that the results match the correlation matrix pretty closely, with L_{avg} being the dominant feature with a 67.5% of importance, which even with its deviation of 14% would still be ahead of the rest. In this case, while the positions of the ranked features are similar to the ones seen before, it seems like most of them are quite

unimportant compared to L_{avg}. From the correlation matrix we expected this for S_{avg} and α_{avg} because of their low correlation to the class, but not for the remaining features.

Figure 4.11 shows an example of two Decision Trees in our Random Forest, where we can see that the nodes are different due to the random selection of features, but that L_{avg} and $N_{AP\text{-}STA}$ appear multiple times in the first levels, while features like RSSI do not (these trees have an average length of 19 levels).



(a) First Decision Tree in Forest



(b) Second Decision Tree in Forest

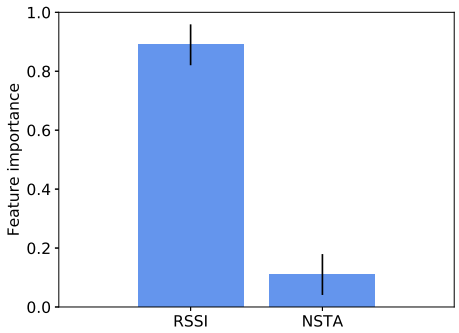
Figure 4.11: First levels of two of the Decision Trees in a Random Forest

Let us test these results. We will now train a Random Forest with different combinations of the features according to what we have learned. Our first idea was that L_{avg} and N_{STA} should be the most important features (we eliminate $N_{AP\text{-}STA}$ due to its strong correlation to L_{avg}). But now we can see that L_{avg} seems to be strong enough on its own. It would also be interesting to see how the rest of the features predict by themselves. Table 4.3 shows the experiments to be performed and their

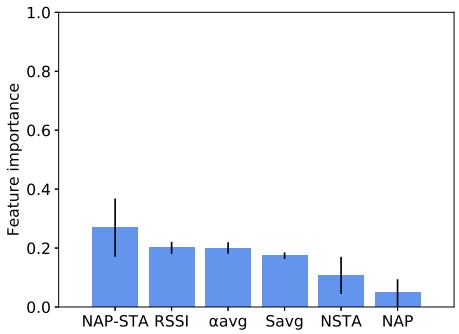
results, with Figure 4.12 showing the feature importance extracted.

Test	**Features**	**Accuracy**
Test 0	All features	97.4%
Test 1	L_{avg}, N_{STA}	99.4%
Test 2	L_{avg}	99.2%
Test 3	All except for L_{avg}	77.9%
Test 4	All except for L_{avg} and $N_{AP\text{-}STA}$	74.09%

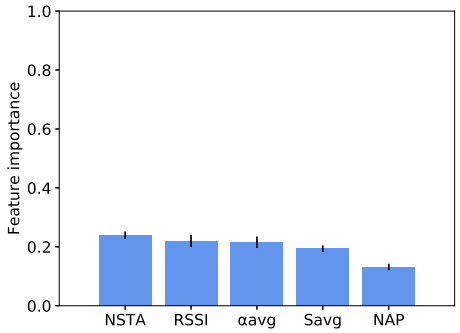
Table 4.3: Tests performed



(a) Test 1



(b) Test 3



(c) Test 4

Figure 4.12: Feature importances for various tests

In regard to the accuracy, Test 1 and 2 outperformed the initial testing with all features, meaning that some of our features are actually detrimental to the classification. Figure 4.12a shows that L_{avg} is still responsible for most of the decision making with an importance of 88.9%, while N_{STA} takes 11.1%. It is remarkable however that Test 1 manages to perform slightly better than Test 2, so even if the importance is not evenly distributed, it seems that N_{STA} can still offer some information. Once we remove L_{avg} we can see that $N_{AP\text{-}STA}$ is not capable of replacing it, with Test 3 achieving 77.9%, and if we remove it we can still obtain a 74.09% accuracy with all other features in Test 4.

4.5 Creating a more complex scenario

One of the reasons for the low importance of several of our features is likely due to the way we created our data set. We used a fixed number of STAs, APs, and all STAs behaved the same. It makes sense then that features like the number of STAs in an AP or the number of APs in the same channel will always be very similar, thus limiting their use in the decision making. We will now create a data set with more variability to test if these features can have a higher level of importance.

We perform 500 simulations again, but now we have two types of users, the first ones will require a throughput in the range of $[1, 8]$ Mbps and have a flow duration exponentially distributed with a mean of 45 seconds, and the others will require from $[8, 16]$ Mbps with a flow duration also exponentially distributed of mean 30 seconds. We also set the number of STAs randomly at the beginning of each simulation, drawing from a range of $[15, 60]$, with the number of APs also being chosen randomly from $[2, 8]$.

We will then check again the correlation and feature importance to see how these features have evolved. In Figure 4.13 we can see that most of the correlation matrix is fairly similar to before, the only major differences are that N_{AP} shows a lower correlation than before with the class, L_{avg} and $N_{AP\text{-}STA}$, and both N_{STA} and $N_{AP\text{-}STA}$ have increased their correlation with the class and between each other. RSSI has lost its correlation to α_{avg}, and the highest correlation is still between L_{avg} and $N_{AP\text{-}STA}$, with it being now at 0.92. It is curious that now $N_{AP\text{-}STA}$ has a stronger correlation to the class than L_{avg}.

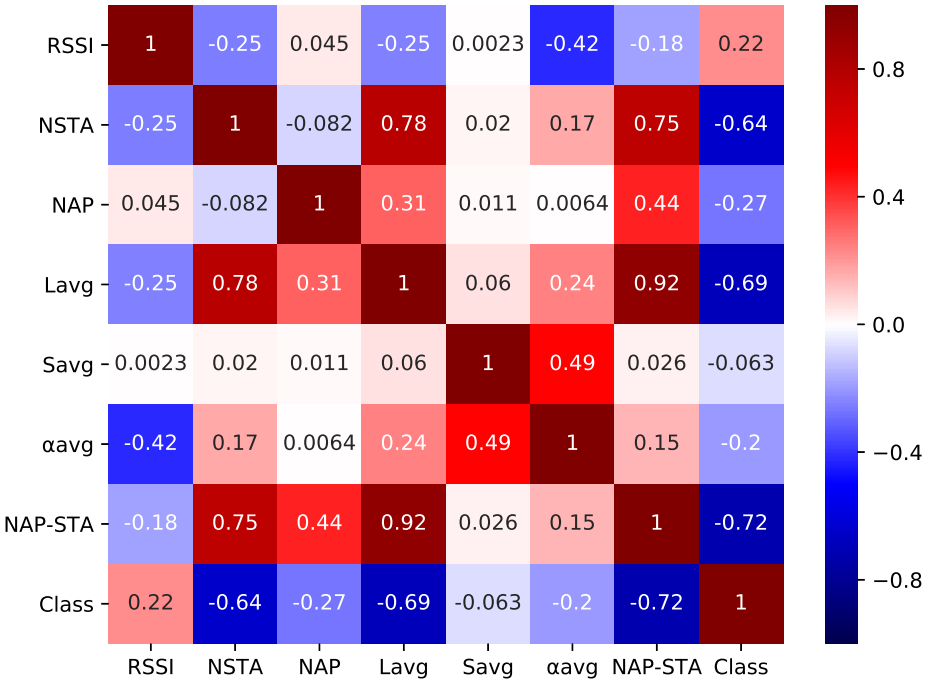


Figure 4.13: Correlation matrix of second data set

Figure 4.14 shows the new feature importance, where the ranking has remained almost the same, but L_{avg} has lost some importance to $N_{AP\text{-}STA}$ and N_{STA}. The biggest difference is the standard deviation, which is now much higher than before, showing that the three first features can swap positions in the forest, while the rest are still not important to the classification.

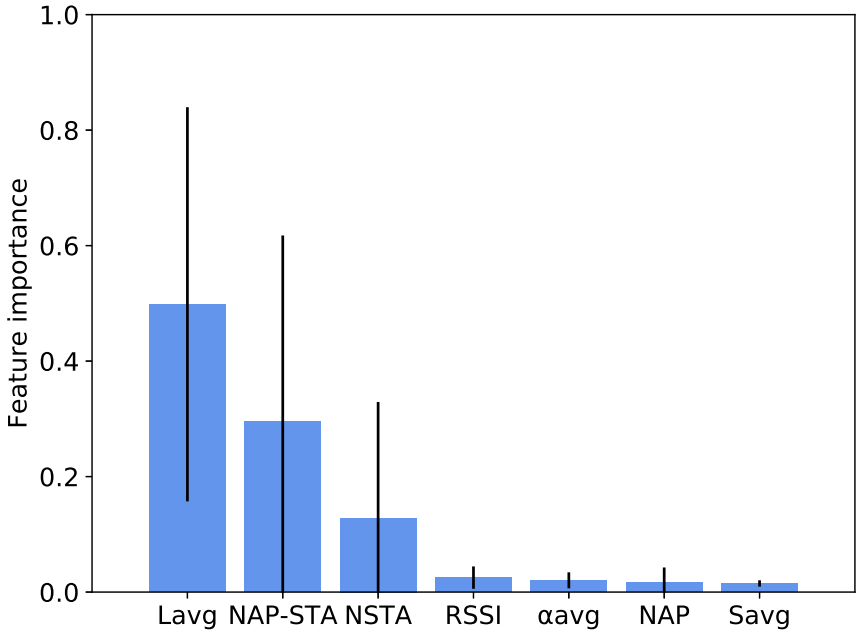


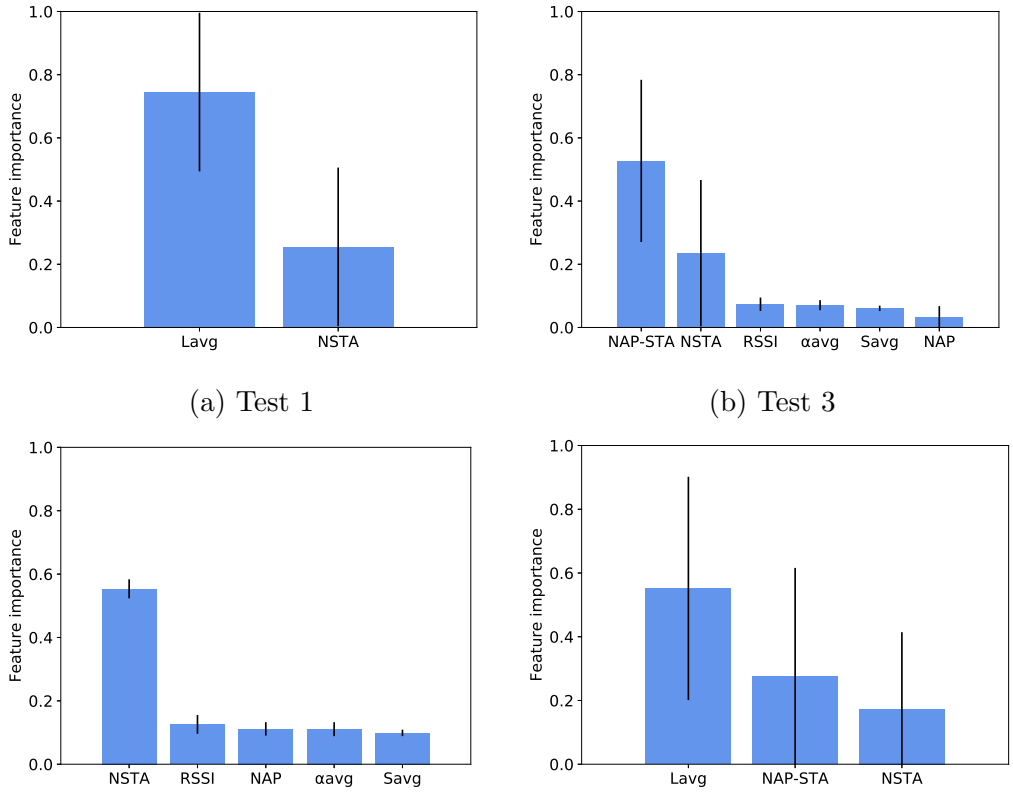
Figure 4.14: Feature importance in second data set

Finally we show the accuracy for the same tests we performed before in Table 4.4, adding a new test where we use the top 3 features, in which we can observe that the decrease in the importance of L_{avg} is clear, with test 3 increasing from 77.9% to 93.9%. We also find that test 0, 1, 2 and 5 have the same performance, which would mean that using only L_{avg} and N_{STA} should be enough now to get the highest performance. Overall, it seems that increasing the variability of the scenario has helped the model perform better than before. High variability means that more information can be extracted from each feature, while in the previous simulation most scenarios were fairly similar, leading to some features, like N_{AP} having always the same value.

Test	**Features**	**Accuracy**
Test 0	All features	98.6%
Test 1	L_{avg}, N_{STA}	98.7%
Test 2	L_{avg}	98.7%
Test 3	All except for L_{avg}	93.9%
Test 4	All except for L_{avg} and $N_{AP\text{-}STA}$	90.5%
Test 5	L_{avg}, N_{STA} and $N_{AP\text{-}STA}$	98.9%

Table 4.4: Tests performed for second data set

We also show the feature importance for each test in Figure 4.15. The importance has shifted in favor of N_{STA}, as it has gone from an 11.1% to 24.1% in test 1, and from 22% to 56% in test 4. It is remarkable that in test 4 the standard deviations are really low compared to the other tests, showing that for that particular group of features N_{STA} will always be at the top.



(a) Test 1

(b) Test 3

(c) Test 4

(d) Test 5

Figure 4.15: Feature importance for second wave of tests

4.6 Validation of the model

As a final validation of the prediction model we will use it in our simulator to predict the satisfaction of the users during the simulation. To do this we create a process in the central controller that will ask for statistics after a certain amount of time since the beginning of the simulation. The AP relays the request to the STAs, collects all information from them, and sends it back to the controller along with its own statistics. Then, the controller does the prediction at a particular interval and we store it until the simulation ends. Then we can compare if the prediction matches the real result.

To use our model we used *sklearn porter*[8] to obtain a C function out of the python

[8]https://github.com/nok/sklearn-porter

model we created. We can then call it in our simulator by putting all the relevant features in an array.

We use 500 simulations with new seeds and test two things: the first one is if our model properly predicts STA satisfaction, and the second one is how much data needs to be aggregated to make a prediction. We want to be able to predict the long term satisfaction with as little data as possible, so we will trigger the prediction mechanism at different intervals to find the optimal one. Figure 4.16 shows the accuracy obtained with each time interval.

Since our simulations last one hour, the first test we do uses 59 minutes (3540 s), meaning that we collect data for 59 minutes, make the prediction and then compare it with the data aggregated over the 60 minutes, obtaining an accuracy of 97.68% with the full feature set, and 97.21% with the reduced set of L_{avg}, N_{STA} and $N_{AP\text{-}STA}$, proving that our model can be used in different scenarios, and correctly predicts the satisfaction for most STAs. Then we keep decreasing the triggering of the mechanism to test how much data we need. We observe that for 60 seconds we only get an 84.36%, and 84.29%, while we already go over 90% on both sets with 120 seconds. At 180 seconds we get 93.04% and 93.05%, and after that the gains are much smaller, as 10 minutes will get us an accuracy of 96.19% and 96.54%, while half an hour gets us 97.25% and 97.14%. Curiously, the accuracy of the smaller feature set outperforms the full set for most intervals, but falls behind a little with the longer intervals of 1800 and 3540 seconds.

In any case, it is clear that both feature sets are pretty much equivalent, and since we want to use the shortest time interval possible, if we use 5 minutes (300 s) we can use the reduced set and obtain an accuracy of 95.11%.

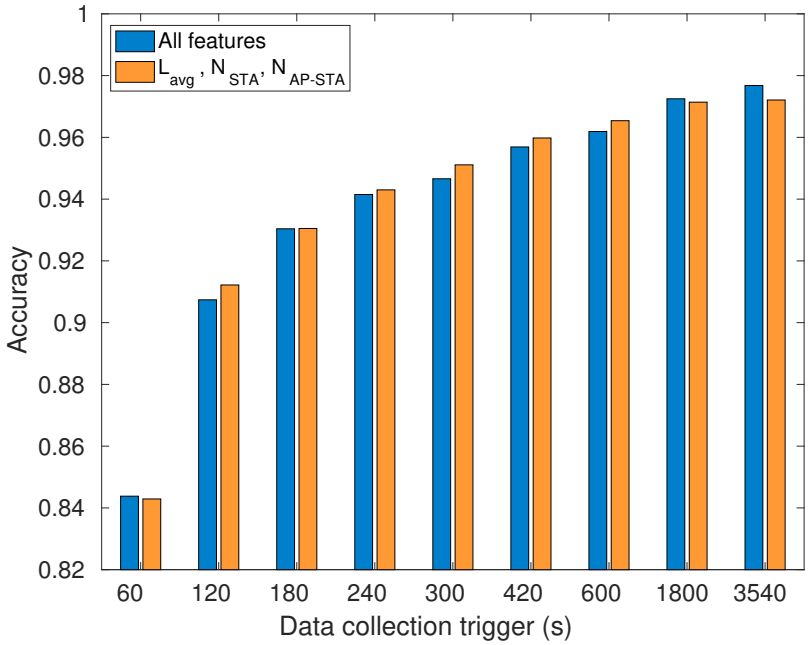


Figure 4.16: Accuracy for each time interval used

Chapter 5

Integration with the simulator

5.1 Introduction

In this Chapter we will create an algorithm in the simulator that uses the prediction model to move STAs that are at risk of being unsatisfied to other APs. We will repeat the same simulations (i.e., use the same scenario and seed) we used during the validation of the model in the previous Chapter, applying the changes live using the data collection trigger of 300 seconds.

5.2 Controller architecture

In this Section we will discuss the amendments that will be considered to enable the functionalities that are implemented in this simulation. We need two things:

1. **Environmental information from the STA:** We need to know which APs are in range of the STA and what is their RSSI. This will be provided by the 802.11k-2008 amendment.
2. **Re-association triggered by the controller:** Re-association parameters will be decided by the central controller, which needs a way to communicate this to the STAs. This can be achieved with 802.11v-2011.

5.2.a 802.11k-2008

This amendment was designed at the same time as 802.11r-2008, with both of these amendments improving user roaming speed. 802.11r allowed APs to share encryption keys so that a STA transitioning from an AP to another in the same network could avoid performing the entire authentication process, while 802.11k defined several methods for exchanging information between a STA and an AP using action frames. Both amendments were incorporated into the 802.11 standard in 2012.

802.11k is most commonly known for its neighbor report, which allows a STA with a low signal to send a request to its AP for information on nearby APs that could be used for roaming. APs then send a response with a list of APs and relevant information, such as their channel, which then allows the STA to save time by avoiding a full scan of all channels. A STA that has received a neighbor report will only scan on channels that are reported to have an AP, greatly shortening the scan time.

Similar to the neighbor report, this amendment defines a beacon report, which works the opposite way. An AP sends a request to the STA for information on the APs that it can sense. The STA then starts a scan (whether passive or active is also defined by the report), and sends back a response to the AP with a full list of sensed APs, channels and the received signal strength. This is the functionality that we will be using to properly inform our AP selection mechanism for each STA.

5.2.b 802.11v-2011

An amendment designed to improve network performance through information sharing between devices, 802.11v defines the Basic Service Set (BSS) transition management to improve roaming functionalities. Also using action frames, APs can send a request if they are overloaded to ask STAs to roam to other APs. STAs can also perform a request for roaming options to the AP if their signal is too low.

For our purposes, we can use a transition request to send the STA a frame containing the ID of nearby APs so that the STA will then re-associate to one of the options in the list. These frames are designed so that the AP list is in order of transition preference, meaning that the first AP in the list is our preferred option for re-association. STAs then will scan according to the priority list, and associate to the first available option. We will use this mechanism so that the controller can re-associate the STAs to other APs according to our algorithm.

5.3 First attempt

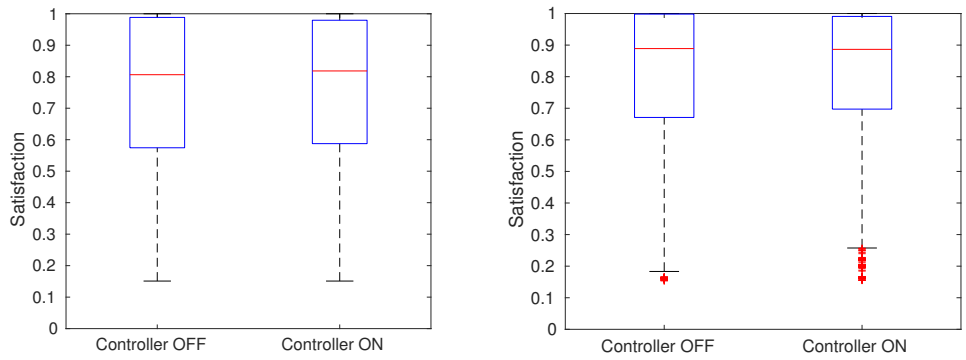
We add a new process to the simulator that is called after the central controller has made a prediction for all available STAs. We also make the STAs send additional information to the controller, like the list of APs that they sensed before associating.

Once all predictions are done by the controller, we count how many unsatisfied STAs we have predicted for every AP, then if there is an AP without negative predictions, we try to move unsatisfied STAs to it. To avoid overloading the unsaturated APs, we only allow one re-association to each satisfied AP. This means that the maximum amount of re-associations per simulation is the number of APs minus 1.

We call on this process once at 300 seconds, and after the re-associations we leave the network as is until the simulation finishes, so that we can see if the overall satisfaction has improved with regards to our baseline (which is the same simulation

without changes). Figure 5.1a shows the average STA satisfaction with and without re-associations, where we find that there is very little difference between the two cases. We do gain some improvement when we re-associate, but overall it looks almost identical. This is due to two main reasons: the first one is that due to the randomness of our simulations, some of the seeds we use lead to completely saturated networks in which the mechanism never triggers due to all APs having negative predictions (which happens in 30.4% of the seeds, or 5383 STAs out of 17611). The second reason is that our re-association limitation limits the amount of moves that can be performed, meaning that the network topology is pretty much the same after the re-associations.

Figure 5.1b shows a subset of the simulation seeds in which re-associations could be applied to the network (meaning that we eliminate those scenarios that remain unchanged), and now we find more differences: we can see that the 75^{th} percentile and the median go down slightly with our method, but the 25^{th} percentile and the minimum increase a fair amount, meaning that we can avoid the worst cases of STA satisfaction with this method. Also remarkable is that the prediction accuracy of our model for these subset of seeds is 93.47% for the baseline and 84.35% with re-associations, meaning that we have modified the final satisfaction of 1606 STAs.



(a) All simulations

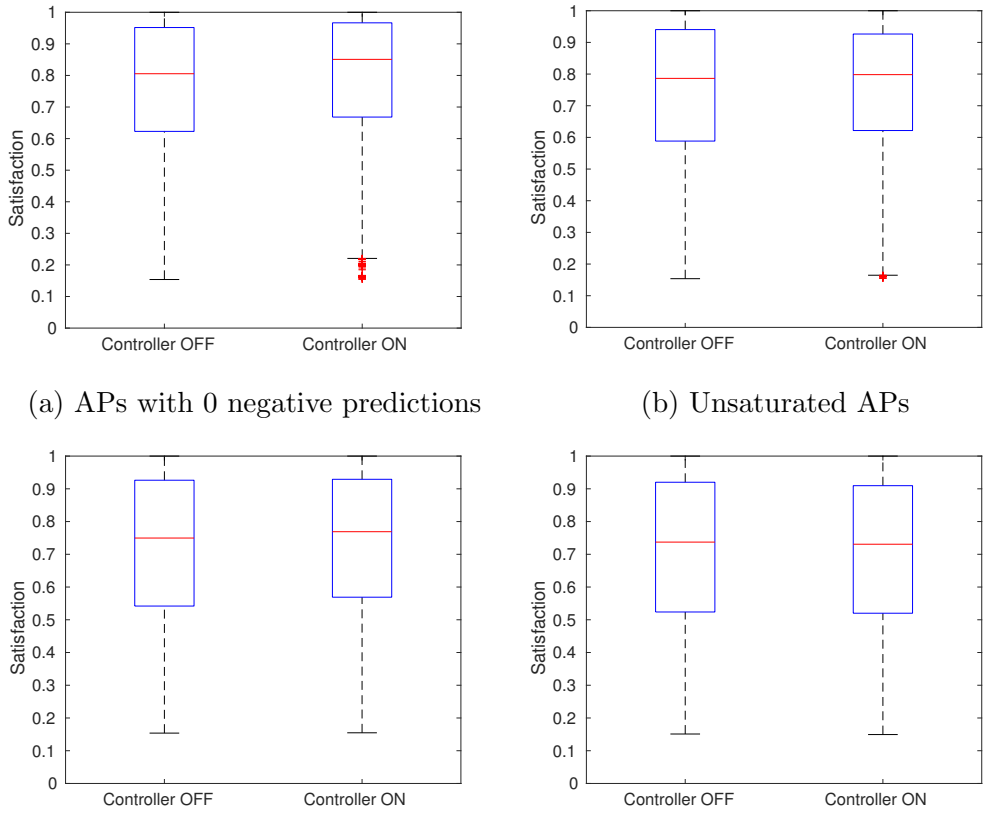
(b) Subset of simulations

Figure 5.1: STA satisfaction with different amounts of simulations used

We will now analyze the efficacy of different methods of finding a suitable AP for re-association, these methods are:

- **No bad predictions:** The method previously used, in which an AP is considered only if no STA associated to it has a negative prediction.
- **Unsaturated APs:** APs with a reported L_{avg} below 1.
- **APs alone in the channel:** APs that do not have other APs in range that share their channel.
- **APs with less bad predictions:** Those APs that have a lower count of STAs with bad predictions than the source AP of the unsatisfied STA.

In Figure 5.2 we show the satisfaction for the unsatisfied STAs (i.e., those with satisfaction < 1 at simulation end). We can observe that the less effective method is using APs with less negative predictions, since Figure 5.2d shows that there is barely any change in the satisfaction of the STAs, aside from the 75^{th} percentile going down, which is not desirable. The other three methods seem to improve the satisfaction of all the troubled STAs, but only using APs with no bad predictions in Figure 5.2a leads to the entire boxplot being raised from its baseline. Using only APs that do not share channel seems to also be very effective, but its minimum remains the same.



(a) APs with 0 negative predictions

(b) Unsaturated APs

(c) APs alone in the channel

(d) APs with less negative predictions

Figure 5.2: Satisfaction of unsatisfied STAs by method

Table 5.1 shows more detailed statistics, with the number of unsatisfied STAs with and without the controller, and the satisfaction achieved by the unsatisfied STAs. All methods increase the number of unsatisfied STAs, but this is not unexpected, as all the methods try to balance the amount of unsatisfied STAs on the network, so our re-associations will always have this side-effect. The redeeming factor is the average satisfaction for these troubled STAs, as we can see that two methods increase their average satisfaction and throughput. The highest increase is for the APs without bad predictions, where we see that the satisfaction increases from 76.42% to 79.37%. Both it and the APs alone in the channel are the only ones that do not decrease the average overall satisfaction, as they keep it at 81% and 75% respectively. From

this we can say that while we do not improve the overall network satisfaction, we do decrease, however slighlty, the gap between satisfied and unsatisfied STAs, leading to a fairer network.

	Controller OFF	**Controller ON**
Method	**Unsatisfied STAs**	**Unsatisfied STAs**
No bad pred.	9756	10731
Unsaturated AP	12465	13528
APs alone in ch.	12852	13488
APs with less bad pred.	14550	15565
Method	**Avg. satisfaction**	**Avg. satisfaction**
No bad pred.	76.42%	79.37%
Unsaturated AP	74.16%	75.33%
APs alone in ch.	71.34%	72.34%
APs with less bad pred.	70.25%	69.85%

Table 5.1: Statistics for unsatisfied STAs

5.4 Improving the method

In the previous section we only attempted a single re-association round. In this section we modify our method to trigger periodically every 5 minutes, obtaining the new information from the modified network, predicting again and attempting more re-associations. We still maintain the same limit to simultaneous re-associations, but now since we will repeat the process multiple times, we get to perform many more re-associations than before.

We want to combine the two methods that offered the most positive results, so first we use the APs with zero bad predictions as a metric repeated every 5 minutes over the first half hour, and then leave the rest of the simulation to remain static. Afterwards we combine the two methods and use only APs that have zero bad predictions and are alone in their channel. For both methods we also add the stipulation that the channel of the new AP has to be different to the channel of the source AP.

Table 5.2 shows the new statistics of unsatisfied STAs for both methods, where we see that the added periodicity does help in improving their effect with both going from 75% to more than 80% satisfaction. For the overall satisfaction of the network, the first method goes from 80.82% to 81.72%, while the second one goes from 80.61% to 82.17%. It seems that combining the two methods is the best approach, but it is also true that the second one is more restrictive, happening only in 319 of 500 seeds (63.8%), while the first method is triggered more frequently, in 370 seeds (74%).

	Controller OFF	Controller ON
Method	**Unsatisfied STAs**	**Unsatisfied STAs**
No bad pred.	10226	12481
No bad pred. + alone in Ch.	8957	10411
Method	**Avg. satisfaction**	**Avg. satisfaction**
No bad pred.	75.83%	81.12%
No bad pred. + alone in Ch.	75.47%	80.59%

Table 5.2: Statistics for unsatisfied STAs with new method

Figure 5.3 shows the boxplots for both methods in comparison to the baseline where the controller is off (we use the baseline of the first method, as it is the higher of the two), and with it we can observe that the first method is the better of the two, as its boxplot reaches higher values. Compared to the baseline without a central controller, it increases the median from 79.8% to 87.4%, the 25$^{\text{th}}$ percentile from 61% to 70%, the minimum from 18.32% to 31.59% and the 75$^{\text{th}}$ percentile from 95.05% to 97.18%.

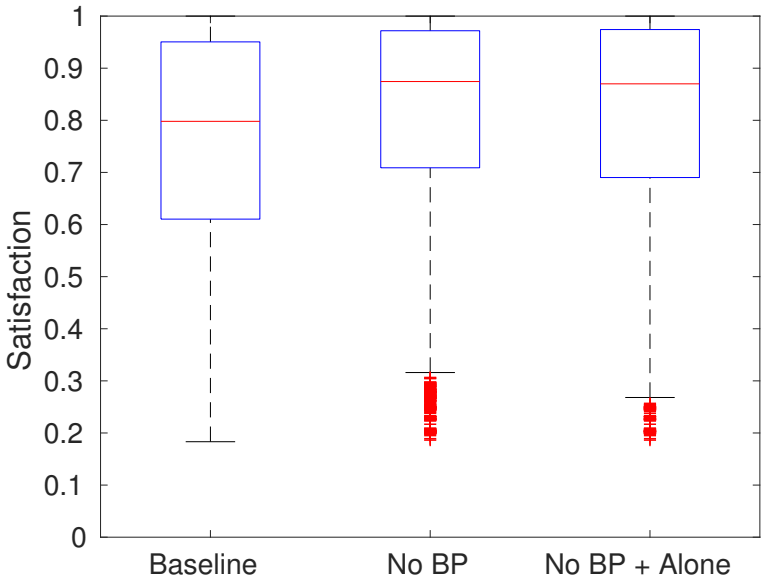


Figure 5.3: Satisfaction of troubled STAs with different methods

Finally, we evaluate the performance of our final method with a single scenario in which we can see how the network is affected. We show the chosen scenario in Figure 5.4, which is a curious case in which the APs closest to each other are on the same channel. Figure 5.5 shows the activity seen by each AP.

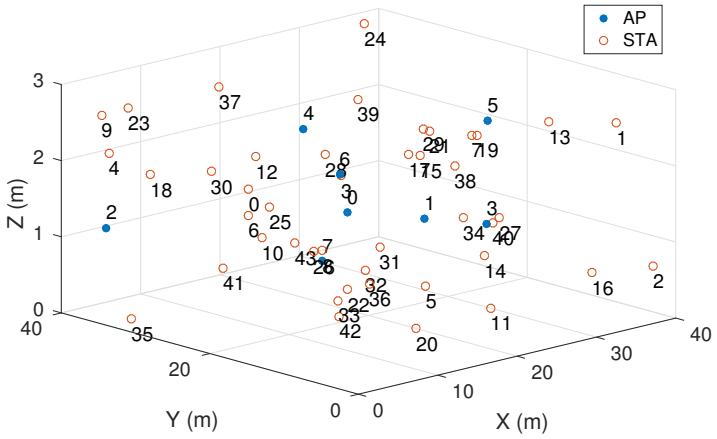


Figure 5.4: Single scenario with 44 STAs and 8 APs

There are 19 re-associations in the whole simulation: in the first interval at 5 minutes three STAs move from AP 0 to AP 1, 5 and 6, and a single STA moves from AP 4 to AP 2. We can see how the load at AP 0 diminishes by almost half after this change, while the others get an increment in their own load. As AP 2, 5 and 6 are in range of each other, this means that they all get the proportional increase of 3 new STAs. At 10 minutes AP 2, 5, 6 and 7 move STAs to AP 4, 0, 3 and 1 respectively. This are very good steerings, as the receiving APs are almost unaffected by the new STAs, while the source APs get a much needed decrease in their load.

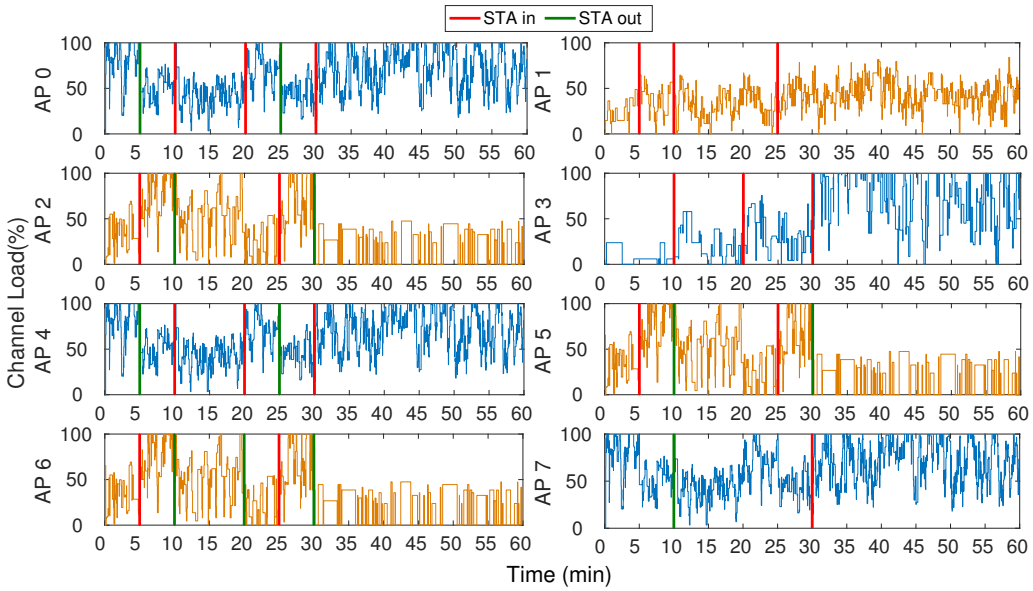


Figure 5.5: Channel load for all APs, APs with same color share channel

At 20 minutes three STAs move from AP 6 to AP 0, 3 and 4. With this, AP 6 halves its load, but AP 0 and AP 4 suffer in the next interval, while AP 3 can deal with the new load. At the 25 minute interval AP 0 and AP 4 both move two STAs

to APs 1, 2, 5 and 6. As before, since APs 2, 5 and 6 share the channel, this does not work well for them, meanwhile AP 1 is still far below saturation.

In the final interval AP 2, 5 and 6 move 4 STAs to AP 0, 3, 4 and 7. AP 0, 4 and 7 share the channel, so these changes are unfavorable, and AP 3 especially suffers in this situation, which we can only assume is due to a really bad RSSI on the new STA. On the other hand, AP 2, 5 and 6 have a really low load for the rest of the simulation.

In general the system acts in the desired way, with the APs with a heavy load moving STAs to those that have a lighter one. We chose this particular case because it shows that our system cannot deal with certain situations, in this case the fact that 6 of the 8 APs are sharing the channel with 2 other APs. As a result we can see the STAs ping ponging between saturated APs. Every once in a while AP 1 and 3 were chosen however, which was the optimal solution, and this is the reason why the unsatisfied STAs achieve higher satisfaction with this method, because the overloaded APs manage to send some of them to the underloaded APs. The optimal solution to this problem however, was to change a couple of APs to the unused channel 11.

5.5 Clustered environments

Up until now we have considered scenarios in which the users are placed randomly following a uniform distribution. In such situations, the strongest RSSI is a good method of association since it spreads the STAs evenly across all APs. We will now consider clustered scenarios in which users form groups of 10 STAs around 5 squared meters. In such situations, STAs tend to overload a single AP and others remain underused.

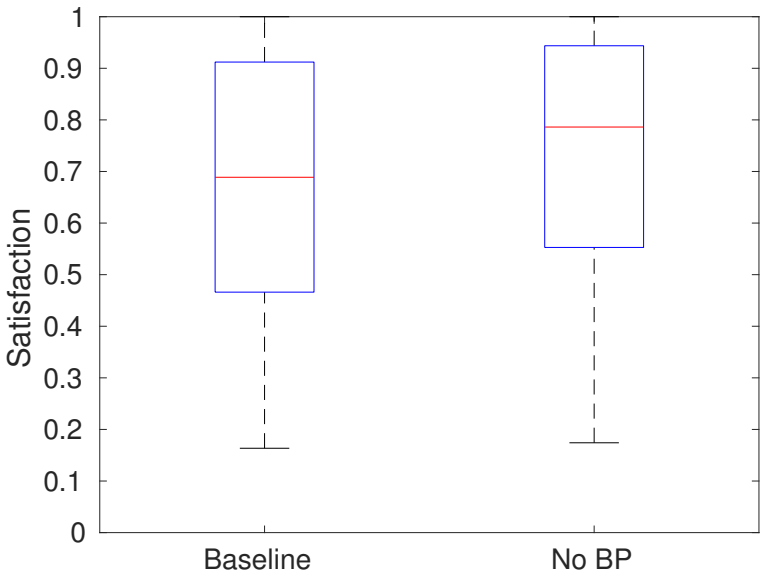


Figure 5.6: Satisfaction of troubled STAs in clustered environments

Figure 5.6 shows the satisfaction for troubled STAs, where we can see a similar behaviour as before, with the entire boxplot raising once we use our algorithm. The 25th percentile raises from 46.6% to 55.2%, the median goes from 68.87% to 78.61%, and the 75th percentile raises from 91.2% to 94.3%.

The average satisfaction for the troubled STAs goes from 67.43% to 72.8%, a 7.9% increase. For the entire network, the satisfaction goes from 71.57% to 73.62%, a 2.8% increase.

We also show a single scenario for the clustered distribution. Figure 5.7 shows the top view of the scenario, where we can see that most users are clustered around AP 0 and AP 4.

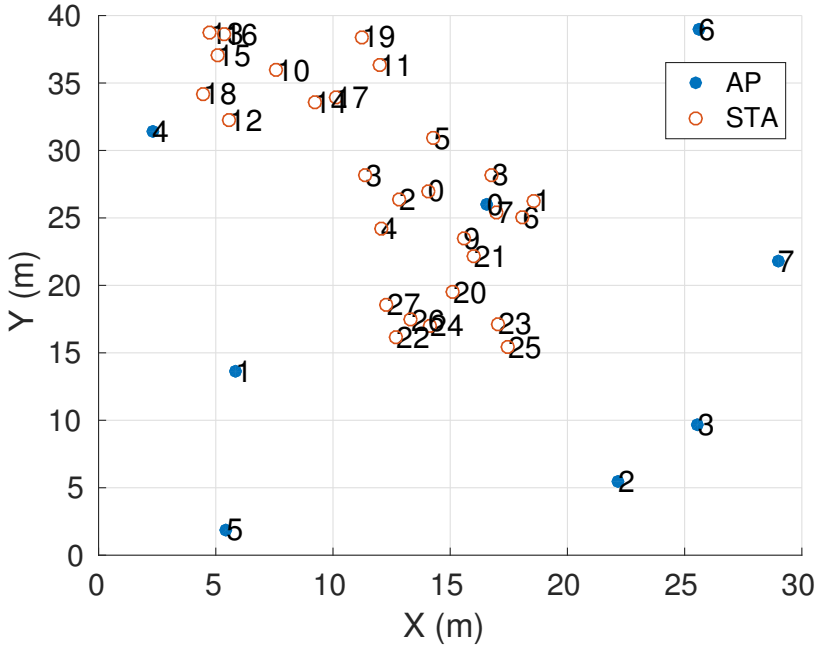


Figure 5.7: Scenario with 8 APs and 28 clustered STAs

Figure 5.8 shows the channel load for the simulation. At 5 minutes AP 0 moves 5 STAs to AP 1, 2, 3, 6 and 7, but we can see it is still saturated due to the high number of STAs associated to it and AP 4. Then at 10 minutes AP 0 moves 5 more STAs to AP 1, 2, 3, 6 and 7 again, and we can see these APs can deal with the new load since they were mostly empty. AP 0 now finally has a more manageable load. At 15 minutes STA 0 moves 4 more STAs, and AP 3 also moves another. These STAs are moved to AP 1, 2, 5, 6 and 7. Now that AP 2 is becoming overloaded, it moves 3 STAs at 20 minutes to APs 0, 1, and 7, and AP 3 also moves one STA to AP 4. Since AP 2 and 3 share the channel, these 4 moves leave them mostly without traffic. At 25 minutes AP 0 and 1 move 2 and 3 STAs each to APs 2, 3, 5, 6 and 7. And finally at 30 minutes AP 4 and 6 move three and one STAs respectively to APs 0, 1 , 3 and 7.

In the final configuration we see that we have managed to take the huge load of AP

0 and 4 and spread it among underused APs like AP 1, 2, 5 and 7, which started the simulation almost without users. We have also seen that in clustered environments we see more re-associations, with AP 0 moving 14 STAs before any other AP needs to do so.

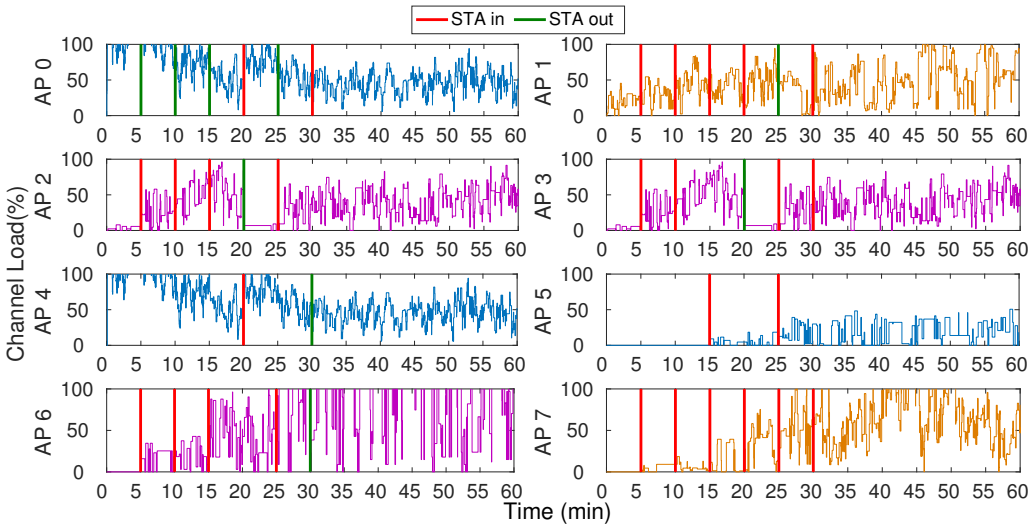


Figure 5.8: Channel load of each AP, APs with same color share channel

Chapter 6

Conclusions

In this work we have studied the use of Supervised Machine Learning for wireless network optimization. We have created a model using Random Forests that can predict if a STA will be satisfied over the course of an hour with an accuracy of 98.9%, trained in networks with different user profiles and network topologies. We have used this model to evaluate which are the best features to be used to identify problems in the network.

We have seen that the commonly used RSSI is not a good measure of the quality of service that a STA will receive, and that the channel load perceived by the AP, as well as the number of STAs in the channel are both much more correlated with the satisfaction of a user. This is due to the channel load implicitly containing a lot of this information, as it is highly correlated to the number of users in the network, and of course, if the users need a lot of throughput this will also be reflected in the channel load.

We then validated the model by introducing it into our simulator, which we have used to test different intervals to find the shortest one with which to make accurate predictions. We then used the number of negative predictions as a way to select candidate APs to which we can steer STAs to improve their service. With this method we balance the number of unsatisfied STAs to other APs, increasing the average satisfaction of troubled STAs by 6.9% and bringing an increase to the minimum satisfaction of 72.4%. Finally, we showed a scenario with the system in place, showing that it manages to offload STAs from overused APs to underused ones.

In the future, it would be interesting to extend the use of our predictions to both improve the current system as well as use it for other mechanisms, such as channel selection or balancing STAs between frequency bands. The current prediction model could also be expanded with the addition of more classes, so as to consider different ranges of user satisfaction, which could then be used for preferential steering of STAs with worse predictions. Further improvements can be made to our simulator as well, such as considering APs and STAs with different 802.11 amendments and adding user mobility.

Bibliography

[1] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper. Technical Report 1486680503328360, February 2019.

[2] Cisco. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper. Technical Report 1551296909190103, February 2019.

[3] B. Bellalta. IEEE 802.11ax: High-efficiency WLANS. *IEEE Wireless Communications*, 23(1):38–46, February 2016.

[4] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo. Machine Learning Paradigms for Next-Generation Wireless Networks. *IEEE Wireless Communications*, 24(2):98–105, April 2017.

[5] M. Bkassiny, Y. Li, and S. K. Jayaweera. A Survey on Machine-Learning Techniques in Cognitive Radios. *IEEE Communications Surveys Tutorials*, 15(3):1136–1159, Third 2013.

[6] Francesc Wilhelmi, Cristina Cano, Gergely Neu, Boris Bellalta, Anders Jonsson, and Sergio Barrachina-Muñoz. Collaborative spatial reuse in wireless networks via selfish multi-armed bandits. *Ad Hoc Networks*, 88:129–141, 2019.

[7] J. Manweiler, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. Predicting length of stay at wifi hotspots. In *2013 Proceedings IEEE INFOCOM*, pages 3102–3110, April 2013.

[8] S. Kajita, H. Yamaguchi, T. Higashino, H. Urayama, M. Yamada, and M. Takai. Throughput and delay estimator for 2.4ghz wifi aps: A machine learning-based approach. In *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 223–226, Oct 2015.

[9] N. Abbas, S. Taleb, H. Hajj, and Z. Dawy. A learning-based approach for network selection in WLAN/3G heterogeneous network. In *2013 Third International Conference on Communications and Information Technology (ICCIT)*, pages 309–313, June 2013.

[10] Magdalena Balazinska and Paul Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-area Network. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 303–316, New York, NY, USA, 2003. ACM.

[11] David Kotz and Kobby Essien. Analysis of a Campus-wide Wireless Network. *Wirel. Netw.*, 11(1-2):115–133, January 2005.

[12] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley. Facilitating access point selection in ieee 802.11 wireless networks. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, pages 26–26, Berkeley, CA, USA, 2005. USENIX Association.

[13] H. Gong, K. Nahm, and J. Kim. Distributed fair access point selection for multi-rate ieee 802.11 wlans. In *2008 5th IEEE Consumer Communications and Networking Conference*, pages 528–532, Jan 2008.

[14] Kaixin Sui, Mengyu Zhou, Dapeng Liu, Minghua Ma, Dan Pei, Youjian Zhao, Zimu Li, and Thomas Moscibroda. Characterizing and improving wifi latency in large-scale operational networks. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 347–360, New York, NY, USA, 2016. ACM.

[15] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang, and X. Qu. Why it takes so long to connect to a wifi access point. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[16] B. Bojovic, N. Baldo, and P. Dini. A Neural Network based cognitive engine for IEEE 802.11 WLAN Access Point selection. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 864–868, Jan 2012.

[17] Marc Carrascosa and Boris Bellalta. Decentralized AP selection using Multi-Armed Bandits: Opportunistic ϵ-Greedy with Stickiness. *CoRR*, abs/1903.00281, 2019.

[18] Task Group AX. Tgax simulation scenarios. Technical report, IEEE P802.11, 2015.